

Smaller SDP for SOS Decomposition

Liyun Dai · Bican Xia

Abstract A popular numerical method to compute SOS (sum of squares of polynomials) decompositions for polynomials is to transform the problem into semi-definite programming (SDP) problems and then solve them by SDP solvers. In this paper, we focus on reducing the sizes of inputs to SDP solvers to improve the efficiency and reliability of those SDP based methods. Two types of polynomials, convex cover polynomials and split polynomials, are defined. A convex cover polynomial or a split polynomial can be decomposed into several smaller sub-polynomials such that the original polynomial is SOS if and only if the sub-polynomials are all SOS. Thus the original SOS problem can be decomposed equivalently into smaller sub-problems. It is proved that convex cover polynomials are split polynomials and it is quite possible that sparse polynomials with many variables are split polynomials, which can be efficiently detected in practice. Some necessary conditions for polynomials to be SOS are also given, which can help refute quickly those polynomials which have no SOS representations so that SDP solvers are not called in this case. All the new results lead to a new SDP based method to compute SOS decompositions, which improves this kind of methods by passing smaller inputs to SDP solvers in some cases. Experiments show that the number of monomials obtained by our program is often smaller than that by other SDP based software, especially for polynomials with many variables and high degrees. Numerical results on various tests are reported to show the performance of our program.

Keywords SOS, SDP, Newton polytope, convex cover polynomial, split polynomial

1 Introduction

Since Hilbert's seventeenth problem was raised in 1900, there has been a lot of work on SOS (sums of squares of polynomials) decomposition. To name a few, please see for instance [1, 30, 34, 19, 13, 23, 24, 2].

Liyun Dai(✉)
LMAM & School of Mathematical Sciences, Peking University
Beijing International Center for Mathematical Research, Peking University
E-mail: dailiyun@pku.edu.cn

Bican Xia
LMAM & School of Mathematical Sciences, Peking University
E-mail: xbc@math.pku.edu.cn

From an algorithmic view of point, writing a multivariate polynomial as an SOS to prove it is non-negative is a crucial part of many applications [38, 13, 23, 11, 31] though the number of non-negative polynomials are much more than the number of sum of squares polynomials [2]. Numerical algorithms for SOS decompositions can handle big scale problems and can be used to get exact results [9]. One main numerical method to solve SOS decomposition problem is to convert it to SDP problem. Actually, there exist some well-known free available SOS solvers which are based on SDP solvers [21, 14, 32].

Obviously, improving SDP solvers' efficiency can improve the efficiency of SDP based SOS solvers. For related work on improving SDP solvers' efficiency, please see for example [20, 17, 35, 41, 27, 37, 16]. It is known that, in the worst case, the size of corresponding SDP problem is $O(\binom{d+n}{d})$ which is polynomial in both n (the number of variables) and d (the degree of given polynomial), if the other one is fixed. In practice, the size of corresponding SDP can be much smaller than $O(\binom{d+n}{d})$ [15, 40, 33]. Although the complexity of SDP is polynomial in d and n , the actual complexity of SDP based SOS solvers are very high since the size of corresponding matrices of SDP is very large when the given polynomial has many variables and high degree. Moreover, the results of existing SDP solvers may be not reliable for large problems [6]. In other words, it is important to reduce the size of corresponding SDP problem so as to improve both the efficiency and reliability of SDP based SOS solvers.

In many practical situations, we do not know more properties of the given polynomial except that the polynomial is sparse, *i.e.*, the number of monomials is much smaller than $\binom{d+n}{d}$. So how to take use of the sparsity to reduce the corresponding size of SDP is a key part to improve the efficiency of solving SOS decomposition problem. For related work on employing sparsity, see for instance [28, 12, 39]. For SOS decomposition of a polynomial on an algebraic variety, a method which may yield smaller SDP is proposed in [25] by combining Gröbner basis techniques with Newton polytope reduction.

In this paper, we focus on reducing the sizes of inputs to SDP solvers to improve the efficiency and reliability of those SDP based methods. Two types of polynomials, convex cover polynomials and split polynomials, are defined. A convex cover polynomial or a split polynomial can be decomposed into several smaller sub-polynomials such that the original polynomial is SOS if and only if the sub-polynomials are all SOS. Thus the original SOS problem can be decomposed equivalently into smaller sub-problems. It is proved that convex cover polynomials are split polynomials and it is quite possible that sparse polynomials with many variables are split polynomials, which can be efficiently detected in practice. Some necessary conditions for polynomials to be SOS are also given, which can help refute quickly those polynomials which have no SOS representations so that SDP solvers are not called in this case. For example, the well-known Motzkin polynomial [18] and Choi-Lam example [4] do not pass the check of the necessary conditions. All the new results lead to a new SDP based method to compute SOS decompositions, which improves this kind of methods by passing smaller inputs to SDP solvers in some cases. Experiments show that the number of monomials obtained by our program is often smaller than that by other SDP based software, especially for polynomials with many variables and high degrees. Numerical results on various tests are reported to show the performance of our program.

The rest part of this paper is organized as follows. Section 2 describes some notations and existing results on SOS, which will be used in this paper. Convex cover polynomial is defined and its property is proved based on the convex property of corresponding Newton polytopes in Section 3. Split polynomial is defined and its property is proved based on monomial relation analysis in Section 4. Moreover, the relationship between the two types of polynomials is given also in Section 4. A new algorithm for SOS decomposition based

on those new results is presented in Section 5. We report some experimental data of our program with comparison to other SDP based tools in Section 6.

2 Preliminary

The symbols $\mathbb{Z}, \mathbb{Z}_+, \mathbb{Q}$ and \mathbb{R} denote the set of integers, natural numbers, rational numbers and real numbers, respectively. If not specified, “polynomials” in this paper are polynomials with real coefficients and are often denoted by p, q, f, g , etc.. By “vectors” we mean vectors in \mathbb{Z}_+^n (or \mathbb{R}^n) which are denoted by α, β, γ , etc.. We use \mathbf{x}, \mathbf{y} denote the variable vectors $(x_1, \dots, x_n), (y_1, \dots, y_n)$, respectively. A hyperplane in \mathbb{R}^n is denoted by $\pi(\mathbf{x}) = 0$.

Consider a polynomial

$$p(\mathbf{x}) = \sum_{\alpha \in \mathbf{P}} c_\alpha \mathbf{x}^\alpha \quad (1)$$

in the variable vector $\mathbf{x} \in \mathbb{R}^n$ with a support $\mathbf{P} \subseteq \mathbb{Z}_+^n$, where $\mathbb{Z}_+ \stackrel{\text{def}}{=} \{x \in \mathbb{Z}, x \geq 0\}$ and real coefficients $c_\alpha \neq 0$ ($\alpha \in \mathbf{P}$). Denote by $S(p)$ the support of a polynomial p . For example, if $p = 1 + x_1^2 + x_2^3$, then $n = 2, S(p) = \{(0, 0), (2, 0), (0, 3)\}$. When $p = 0$, define $S(p) = \emptyset$.

For any $T \subseteq \mathbb{R}^n$ and $k \in \mathbb{R}$, denote by kT the set $\{k\alpha \mid \alpha \in T\}$, where $k(a_1, \dots, a_n) = (ka_1, \dots, ka_n)$, and by $\text{conv}(T)$ the convex hull of T . Let \mathbf{P}^e be the set of $\alpha \in \mathbf{P}$ whose coordinates α_k ($k = 1, 2, \dots, n$) are all even non-negative integers, i.e., $\mathbf{P}^e = \mathbf{P} \cap (2\mathbb{Z}_+^n)$.

Obviously, $p(\mathbf{x})$ can be represented in terms of a sum of squares of polynomials or in short, p is SOS, if and only if there exist polynomials $q_1(\mathbf{x}), \dots, q_s(\mathbf{x}) \in \mathbb{R}[\mathbf{x}]$ such that

$$p(\mathbf{x}) = \sum_{i=1}^s q_i(\mathbf{x})^2. \quad (2)$$

To find both s and polynomials $q_1(\mathbf{x}), \dots, q_s(\mathbf{x})$, it is necessary to estimate and decide the supports of unknown polynomials $q_i(\mathbf{x})$ ($i = 1, \dots, s$). Let \mathbf{Q}_i be an unknown support of the polynomial $q_i(\mathbf{x})$ ($i = 1, \dots, s$). Then each polynomial $q_i(\mathbf{x})$ is represented as

$$q_i(\mathbf{x}) = \sum_{\alpha \in \mathbf{Q}_i} c_{(i,\alpha)} \mathbf{x}^\alpha \quad (3)$$

with nonzero coefficients $c_{(i,\alpha)}$ ($\alpha \in \mathbf{Q}_i, i = 1, \dots, s$).

Suppose $p(\mathbf{x})$ is of the form (2), then $\mathbf{P} \subseteq \text{conv}(\mathbf{P}^e)$ [28]. The following relation is also known by [28]:

$$\{\alpha \in \mathbb{Z}_+^n : \alpha \in \mathbf{Q}_i \text{ and } c_{(i,\alpha)} \neq 0 \text{ for some } i \in \{1, 2, \dots, s\}\} \subseteq \frac{1}{2} \text{conv}(\mathbf{P}^e). \quad (4)$$

Hence we can confine effective supports of unknown polynomials $q_1(\mathbf{x}), \dots, q_s(\mathbf{x})$ to subsets of

$$\mathbf{Q}^0 = \left(\frac{1}{2} \text{conv}(\mathbf{P}^e) \right) \cap \mathbb{Z}_+^n. \quad (5)$$

Definition 1 For a polynomial p , a set $\mathbf{Q} \subseteq \mathbb{Z}_+^n$ is said to satisfy the relation $\text{SOSS}(p, \mathbf{Q})$ (SOSS stands for SOS support) if

$$p \text{ is SOS} \implies \text{there exist } q_i (i = 1, \dots, s) \text{ such that } p = \sum_{i=1}^s q_i^2 \text{ and } S(q_i) \subseteq \mathbf{Q}.$$

For every given p , the problem is how to find a small \mathbf{Q} such that $\text{SOSS}(p, \mathbf{Q})$ holds, *i.e.*, prune more unnecessary monomials from the decomposition. In general, one can start from a coarse \mathbf{Q} , keep eliminating elements of \mathbf{Q} which does not satisfy some conditions, and finally obtain a smaller \mathbf{Q} . Obviously, \mathbf{Q}^0 of (5) satisfies $\text{SOSS}(p, \mathbf{Q}^0)$ for every given p . If q_i satisfies (2), the relation $\text{SOSS}(p, \cup_i \mathbf{S}(q_i))$ holds.

There are two possible approaches to construct q_1, \dots, q_s .

One approach assumes polynomials q_1, \dots, q_s do not share common support. Then each polynomial $q_i(\mathbf{x})$ is represented as Eq. (3). Unfortunately, it is difficult to find exact \mathbf{Q}_i if we do not know more information of p . But when p is correlatively sparse, a correlative sparsity pattern graph is defined in [39] to find a certain sparse structure in p . And this structure can be used to decide different relaxed \mathbf{Q}_i . Theoretically, the relaxations in [39] are not guaranteed to generate lower bounds with the same quality as those generated by the original SOS representation.

The other approach assumes that all polynomials $q_1(\mathbf{x}), \dots, q_s(\mathbf{x})$ share a common unknown support $\mathbf{Q} \subseteq \mathbf{Q}^0$ and each polynomial $q_i(\mathbf{x})$ is represented as

$$q_i(\mathbf{x}) = \sum_{\alpha \in \mathbf{Q}} c_{(i,\alpha)} \mathbf{x}^\alpha. \quad (6)$$

Then Eq. (2) is equivalent to the existence of a positive semi-definite matrix M such that

$$p(\mathbf{x}) = \mathbf{Q}^T(\mathbf{x}) M \mathbf{Q}(\mathbf{x}), \quad (7)$$

where $\mathbf{Q}(\mathbf{x})$ is a vector of monomials corresponding to the support \mathbf{Q} . So in the view of practical computing, finding the SOS representation is equivalent to solving the feasibility problem of (7). Thus, the original problem can be solved by SDP solvers. This approach was presented in [21, 11, 14, 32]. There are close connections between SOS polynomials and positive semi-definite matrices [5, 26, 22, 13].

Notation 1 We denote by $\text{SOS}(p, \mathbf{Q})$ an algorithm of finding positive semi-definite matrix M with \mathbf{Q} under constraints (7).

Let us give a rough complexity analysis of $\text{SOS}(p, \mathbf{Q})$. Let $n = \#(\mathbf{Q})$, the number of elements contained in \mathbf{Q} . Then the size of matrix M in (7) is $n \times n$. Let m be the number of different elements occurring in $\mathbf{Q}\mathbf{Q}^T$. It is easy to know $n \leq m \leq n^2$. Suppose $m = O(n^c)$, $c \in [1, 2]$ and we use *interior point method* in $\text{SOS}(p, \mathbf{Q})$, which is a main method for solving SDP. Then the algorithm will repeatedly solve *least squares* problems with m linear constraints and $\frac{(n+1)n}{2}$ unknowns. Suppose that the least squares procedure is called k times. Then, the total complexity is $O(kn^{2+2c})$. So, if n becomes $2n$, the time consumed will increase by at least 16 times. So reducing \mathbf{Q} 's size is a key point to improve such algorithms.

3 Convex cover polynomial

We give a short description of Newton polytope in Section 3.1. In Section 3.2, we first prove a necessary condition (Theorem 1) for a polynomial to be SOS based on the properties of Newton polytope. Then a new concept, *convex cover polynomial* (Definition 3), is introduced, which leads to the main result (Theorem 2) of this section, that is, a convex cover polynomial is SOS if and only if some smaller polynomials are SOS.

3.1 Newton polytope

Newton polytope is a classic tool. We only introduce some necessary notations here. For formal definitions of the concepts, please see for example [36]. A *polytope* is a subset of \mathbb{R}^n that is the convex hull of a finite set of points. A simple example is the convex hull of $\{(0,0,0), (0,1,0), (0,0,1), (0,1,1), (1,0,0), (1,1,0), (1,0,1), (1,1,1)\}$ in \mathbb{R}^3 ; this is the regular 3-cube. A d -dimensional polytope has many *faces*, which are again polytopes of various dimensions from 0 to $d-1$. The 0-dimensional faces are called *vertices*, the 1-dimensional faces are called *edges*, and the $(d-1)$ -dimensional faces are called *facets*. For instance, the cube has 8 vertices, 12 edges, and 6 facets. If $d=2$ then the edges coincide with the facets. A 2-dimensional polytope is called a *polygon*.

For a given polynomial p , each term $\mathbf{x}^\alpha = x_1^{\alpha_1} \cdots x_n^{\alpha_n}$ appearing in p corresponds to an integer lattice point $(\alpha_1, \dots, \alpha_n)$ in \mathbb{R}^n . The convex hull of all these points (called the *support* of p) is defined as *Newton polytope* of p and is denoted by

$$\mathbf{N}(p) \stackrel{\text{def}}{=} \text{conv}(\mathbf{S}(p)).$$

Definition 2 For a polynomial $p = \sum_{\alpha} c_{\alpha} \mathbf{x}^{\alpha}$ and a set $T \subseteq \mathbb{R}^n$, denote by $\text{Proj}(p, T)$ the polynomial obtained by deleting the terms $c_{\alpha} \mathbf{x}^{\alpha}$ of p where $\alpha \notin (T \cap \mathbb{Z}_+^n)$.

Example 1 Let $p = 2x_1^4 + 4x_2^4 - 3x_3^2 + 1$ and $T = \{(0,0,0), (1,0,0), (4,0,0)\}$, then $\text{Proj}(p, T) = 2x_1^4 + 1$.

3.2 Convex cover polynomial

We guess that Theorem 1 in this section should be a known result. However, we do not find a proof in the literature. So, we prove it here. Since the results of the following Lemma 1 are either obvious or known, we omit the proofs.

Lemma 1 • For any two polynomials f, g , two real numbers k_1, k_2 and any $T \subseteq \mathbb{Z}_+^n$,

$$\text{Proj}(k_1 f + k_2 g, T) = k_1 \text{Proj}(f, T) + k_2 \text{Proj}(g, T).$$

- For any $T \subseteq \mathbb{Z}_+^n$ and any $k \in \mathbb{R} \setminus \{0\}$, we have $k(\frac{1}{k}T \cap \mathbb{Z}_+^n) \subseteq T$.
- Suppose N is an n -dimensional polytope. For any face F of N , there is an $(n-1)$ -dimensional hyperplane $\pi(\mathbf{y}) = 0$ such that $\pi(\alpha) = 0$ for any $\alpha \in F$ and $\pi(\beta) > 0$ for any $\beta \in N \setminus F$.
- Suppose $\pi(\mathbf{y}) = 0$ is a hyperplane and $F \subseteq \mathbb{Z}_+^n \cap (\pi(\mathbf{y}) = 0)$. For any polynomial $p = \sum_{\alpha} c_{\alpha} \mathbf{x}^{\alpha}$ in n variables, we have

$$\mathbf{S}(\text{Proj}(p, F)) \subseteq \mathbf{S}(\text{Proj}(p, \mathbf{S}(p) \cap (\pi(\mathbf{y}) = 0))).$$

- If f, g are two polynomials and $\mathbf{S}(f) \cap \mathbf{S}(g) = \emptyset$, then $\mathbf{S}(f+g) = \mathbf{S}(f) \cup \mathbf{S}(g)$.
- Let $T_1 = \mathbf{S}(f)$ and $T_2 = \mathbf{S}(g)$ for two polynomials f and g . Then $\mathbf{S}(fg) \subseteq T_1 + T_2$, where $T_1 + T_2$ is the Minkowski sum of T_1 and T_2 .

Lemma 2 Suppose $\pi(\mathbf{y}) = 0$ is a hyperplane, $T \subseteq \mathbb{Z}_+^n$ and f, g are two n -variate polynomials. Let $T_1 = \mathbf{S}(f)$, $T_2 = \mathbf{S}(g)$. If $T \subseteq \{\mathbf{y} \mid \pi(\mathbf{y}) = 0\}$, $2T_1 \subseteq \{\mathbf{y} \mid \pi(\mathbf{y}) \geq 0\}$ and $2T_2 \subseteq \{\mathbf{y} \mid \pi(\mathbf{y}) > 0\}$, then $\text{Proj}(fg, T) = 0$.

Proof By Lemma 1, $S(fg) \subseteq T_1 + T_2$. By the definition of Minkowski sum, for any $\alpha \in T_1 + T_2$ there exist $\alpha_1 \in T_1, \alpha_2 \in T_2$ such that $\alpha = \alpha_1 + \alpha_2$. Because $\pi(2\alpha_1) \geq 0$ and $\pi(2\alpha_2) > 0$, $\pi(\alpha) = \pi(\alpha_1 + \alpha_2) = \frac{1}{2}(\pi(2\alpha_1) + \pi(2\alpha_2)) > 0$. So $T_1 + T_2 \subseteq \{\mathbf{y} \mid \pi(\mathbf{y}) > 0\}$. Thus, $S(fg) \cap (\pi(\mathbf{y}) = 0) = \emptyset$ which implies $\text{Proj}(fg, T) = 0$ by Lemma 1 and $T \subseteq (\pi(\mathbf{y}) = 0)$. \square

Lemma 3 Suppose $p = \sum_{i=1}^s q_i^2$ and F is a face of $\mathcal{N}(p)$. Let $F_z = F \cap \mathbb{Z}_+^n, F_{\frac{z}{2}} = \frac{1}{2}F \cap \mathbb{Z}_+^n, q'_i = \text{Proj}(q_i, F_{\frac{z}{2}}), q''_i = q_i - q'_i, T'_i = S(q'_i)$ and $T''_i = S(q''_i)$, then there is a hyperplane $\pi(\mathbf{y}) = 0$ such that

- (1) $F \subseteq \{\mathbf{y} \mid \pi(\mathbf{y}) = 0\}$,
- (2) $2T'_i \subseteq \{\mathbf{y} \mid \pi(\mathbf{y}) = 0\}$, and
- (3) $2T''_i \subseteq \{\mathbf{y} \mid \pi(\mathbf{y}) > 0\}$.

Proof By Lemma 1, there is a hyperplane $\pi(\mathbf{y}) = 0$ such that $\forall \alpha \in F, \pi(\alpha) = 0$ and $\forall \alpha \in \mathcal{N}(p) \setminus F, \pi(\alpha) > 0$. We prove that π is a hyperplane which satisfies the requirement. First, because $T'_i \subseteq F_{\frac{z}{2}}$, by Lemma 1, $2T'_i \subseteq 2F_{\frac{z}{2}} \subseteq F$ and thus $2T'_i \subseteq \{\mathbf{y} \mid \pi(\mathbf{y}) = 0\}$.

Second, it is obvious that $T''_i \cap F_{\frac{z}{2}} = \emptyset, T'_i \cap T''_i = \emptyset$ and $T'_i \cup T''_i = T_i$ where $T_i = S(q_i)$. By Equation (4), we have $T_i \subseteq \frac{1}{2}\mathcal{N}(p)$ and $2T_i \subseteq \mathcal{N}(p)$. Thus $2T''_i \subseteq 2T_i \subseteq \mathcal{N}(p) \subseteq \{\mathbf{y} \mid \pi(\mathbf{y}) \geq 0\}$. If there is an $\alpha \in T''_i$ such that $\pi(2\alpha) = 0$, then $\alpha \in F_{\frac{z}{2}}$, which contradicts with $T''_i \cap F_{\frac{z}{2}} = \emptyset$. Therefore, $2T''_i \subseteq \{\mathbf{y} \mid \pi(\mathbf{y}) > 0\}$. \square

Using the above lemmas, we prove Theorem 1 now.

Theorem 1 If p is SOS, then $\text{Proj}(p, F)$ is SOS for every face F of $\mathcal{N}(p)$.

Proof Suppose $p = \sum_{i=1}^s q_i^2$ and F is a face of $\mathcal{N}(p)$. Let $F_z = F \cap \mathbb{Z}_+^n, q'_i = \text{Proj}(q_i, \frac{1}{2}F_z)$ and $q''_i = q_i - q'_i$. Then $p = \sum_{i=1}^s (q'_i + q''_i)^2 = \sum_{i=1}^s q_i'^2 + 2 \sum_{i=1}^s q'_i q''_i + \sum_{i=1}^s q_i''^2$. By Lemma 1, $\text{Proj}(p, F_z) = \sum_{i=1}^s \text{Proj}(q_i'^2, F_z) + 2 \sum_{i=1}^s \text{Proj}(q'_i q''_i, F_z) + \sum_{i=1}^s \text{Proj}(q_i''^2, F_z)$. By Lemma 3, there is a hyperplane $\pi(\mathbf{y}) = 0$ such that (1) $\forall \alpha \in F, \pi(\alpha) = 0$; (2) $\forall \alpha \in \mathcal{N}(p) \setminus F, \pi(\alpha) > 0$; (3) for any $q'_i, 2S(q'_i) \subseteq \{\mathbf{y} \mid \pi(\mathbf{y}) = 0\}$; and (4) $2S(q''_i) \subseteq \{\mathbf{y} \mid \pi(\mathbf{y}) > 0\}$. By Lemma 2, $\text{Proj}(q'_i q''_i, F_z) = 0$ and $\text{Proj}(q_i''^2, F_z) = 0$. Therefore, the intersect between support of $2 \sum_{i=1}^s q'_i q''_i + \sum_{i=1}^s q_i''^2$ and F_z is an emptyset, i.e., $\text{Proj}(p, F) = \text{Proj}(p, F_z) = \sum_{i=1}^s \text{Proj}(q_i'^2, F_z) = \sum_{i=1}^s (q_i')^2$. The last equality holds because $S(q_i'^2) \subseteq F_z$. \square

Remark 1 Theorem 1 is strongly related to Theorem 3.6 of [29], which states that if p is positive semidefinite, then $\text{Proj}(p, F)$ is positive semidefinite for every face F of $\mathcal{N}(p)$.

Theorem 1 proposes a necessary condition for a polynomial to be SOS.

Example 2 $p = x_1^4 + x_2^4 + x_3^4 - 1$.

Obviously, the polynomial in Example 2 is not SOS (e.g., $p(0, 0, 0) = -1$). By Theorem 1, one necessary condition for p to be SOS is that $\text{Proj}(p, \{(0, 0, 0)\}) = -1$ should be SOS which can be efficiently checked. On the other hand, if we use Newton polytope based method to construct \mathbf{Q} in (7), the size of \mathbf{Q} is $\binom{3+2}{2} = 10$. The number of constraints is $\binom{3+4}{4} = 35$.

Definition 3 (Convex cover polynomial) A polynomial p is said to be a *convex cover polynomial* if there exist some pairwise disjoint faces $F_i (i = 1, \dots, u)$ of $\mathcal{N}(p)$ such that $S(p) \subseteq \cup_{i=1}^u F_i$.

It is easy to get the following proposition by the definition of convex cover polynomial.

Proposition 1 *The support of a convex cover polynomial does not intersect the interior of its Newton polytope.*

The following theorem is a direct corollary of Theorem 1.

Theorem 2 *Suppose p is a convex cover polynomial and $F_i (i = 1, \dots, u)$ are the faces satisfying the condition of Definition 3. Let $p_i = \text{Proj}(p, F_i) (i = 1, \dots, u)$. Then p is SOS if and only if p_i is SOS for $i = 1, \dots, u$.*

We use the following example to demonstrate the benefit of Theorem 2.

Example 3 $p = x_1^6 + x_2^6 + x_1^4 - 2x_1^2x_2^2 + x_2^4$.

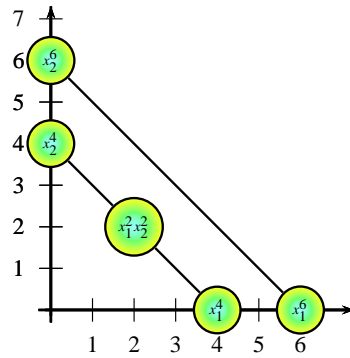


Fig. 1 Newton polytope of Example 3.

For Example 3, $\mathcal{S}(p) = \{(6, 0), (0, 6), (4, 0), (2, 2), (0, 4)\}$. Let $F_1 = \{(6, 0)\}$, $F_2 = \{(0, 6)\}$, $F_3 = \text{conv}(\{(4, 0), (2, 2), (0, 4)\})$ (shown in Fig. 1) be three faces of $\mathcal{N}(p)$. Because F_1, F_2, F_3 satisfy the condition of Definition 3, p is a convex cover polynomial. Let $p_i = \text{Proj}(p, F_i)$ for $i = 1, 2, 3$. Then, by Theorem 2, proving p is SOS is equivalent to proving p_i is SOS for $i = 1, 2, 3$. Therefore, the original problem is divided into three simpler sub-problems. When using Newton polytope based method to prove p is SOS, the size of \mathbf{Q} is 7 and the number of constraints is 18. However, for p_1, p_2, p_3 , the corresponding data are $(1, 1)$, $(1, 1)$ and $(3, 5)$, respectively.

Dividing the original problem into simpler sub-problems can improve not only the efficiency but also the reliability of the results. As indicated in [6], when the scale of problem is large, the numerical error of SDP solver may lead to a result which looks like “reliable” by the output data while it is indeed unreliable.

4 Split polynomial

Another new concept, *split polynomial*, is introduced in this section. Every convex cover polynomial is a split polynomial. The main results of this section are Theorem 3 (analogue of Theorem 1) and Theorem 4 (analogue of Theorem 2), which allow one to block-diagonalize a wider class of SDPs.

Definition 4 For a set \mathbf{Q} of vectors and any $\alpha \in \mathbf{Q} + \mathbf{Q}$, define $\varphi_{\mathbf{Q}}(\alpha) = \{\beta \in \mathbf{Q} \mid \exists \gamma \in \mathbf{Q}, \beta + \gamma = \alpha\}$.

Definition 5 Suppose \mathbf{Q} satisfies $SOSS(p, \mathbf{Q})$ (see Definition 1) for a polynomial p . Define $\mathcal{V}(p, \mathbf{Q})$ to be the set $\{\alpha \in \mathbf{Q} \mid \varphi_{\mathbf{Q}}(2\alpha) = \{\alpha\}\}$.

Definition 6 Suppose \mathbf{Q} satisfies $SOSS(p, \mathbf{Q})$ for a polynomial p . For any $\alpha \in \mathbf{Q} + \mathbf{Q}$, define

$$\psi_{\mathbf{Q}}(\alpha) = \begin{cases} \{\frac{1}{2}\alpha\} & \text{if } \varphi_{\mathbf{Q}}(\alpha) = \{\frac{1}{2}\alpha\}, \\ \bigcup_{\beta, \gamma \in \mathbf{Q}, \beta \neq \gamma, \beta + \gamma = \alpha} (\psi_{\mathbf{Q}}(2\beta) \cup \psi_{\mathbf{Q}}(2\gamma)) & \text{otherwise.} \end{cases}$$

If $\alpha \notin \mathbf{Q} + \mathbf{Q}$, define $\psi_{\mathbf{Q}}(\alpha) = \emptyset$.

Since $\psi_{\mathbf{Q}}(\alpha)$ is a subset of $\mathcal{V}(p, \mathbf{Q})$ and obviously $\mathcal{V}(p, \mathbf{Q})$ is a finite set, Definition 6 makes sense.

Lemma 4 Suppose \mathbf{Q} satisfies $SOSS(p, \mathbf{Q})$ for a polynomial p and F is a face of $\text{conv}(\mathbf{Q} + \mathbf{Q})$. Let $T = \{\alpha \mid \alpha \in \mathcal{V}(p, \mathbf{Q}), 2\alpha \in F\}$, $\mathbf{Q}_1 = (\mathbf{Q} + \mathbf{Q}) \cap F$. Then $\psi_{\mathbf{Q}}(\alpha) \subseteq T$ for any $\alpha \in \mathbf{Q}_1$.

Proof For any $\beta \in \psi_{\mathbf{Q}}(\alpha)$, by the definition of $\psi_{\mathbf{Q}}(\alpha)$, there are $\beta_1, \dots, \beta_k, \gamma_1, \dots, \gamma_k \in \mathbf{Q}$ such that $\beta_i \neq \gamma_i$ for $i = 1, \dots, k-1$, $\alpha = \beta_1 + \gamma_1$, $2\beta_1 = \beta_2 + \gamma_2, \dots, 2\beta_{k-1} = \beta_k + \gamma_k$, $\beta_k = \gamma_k = \beta$ and $\psi_{\mathbf{Q}}(2\beta) = \{\beta\}$.

We prove $2\beta_i \in F$ by induction. Because $\alpha = \beta_1 + \gamma_1$ and $\alpha \in F$, we have $2\beta_1 \in F$. Assume that $2\beta_i \in F$ for $i < m$. If $i = m$, since $2\beta_{m-1} = \beta_m + \gamma_m$ and $2\beta_{m-1} \in F$, we have $2\beta_m \in F$. Then $2\beta = 2\beta_k \in F$ and hence, $\beta \in T$. \square

Definition 7 Suppose \mathbf{Q} satisfies $SOSS(p, \mathbf{Q})$ for a polynomial p and $T \subseteq \mathcal{V}(p, \mathbf{Q})$. Define $\sigma(T) = \{\gamma \mid \gamma \in \mathbf{Q}, \psi_{\mathbf{Q}}(2\gamma) \subseteq T\}$.

Lemma 5 Suppose p is SOS, say $p = \sum_{i=1}^s h_i^2$, and $\mathcal{S}(h_i) \subseteq \mathbf{Q}$. Then for any $T \subseteq \mathcal{V}(p, \mathbf{Q})$ and any $\beta \in \mathbf{Q} + \mathbf{Q}$, if $\psi_{\mathbf{Q}}(\beta) \subseteq T$, then $\beta \notin \mathcal{S}((p - \sum_{i=1}^s \text{Proj}(h_i, \sigma(T))^2))$.

Proof For any $\gamma_1, \gamma_2 \in \mathbf{Q}$ with $\gamma_1 + \gamma_2 = \beta$, we have $\psi_{\mathbf{Q}}(2\gamma_1) \subseteq \psi_{\mathbf{Q}}(\beta)$ and $\psi_{\mathbf{Q}}(2\gamma_2) \subseteq \psi_{\mathbf{Q}}(\beta)$ by the definition of $\psi_{\mathbf{Q}}$. Since $\psi_{\mathbf{Q}}(\beta) \subseteq T$, we have $\gamma_1, \gamma_2 \in \sigma(T)$ by the definition of $\sigma(T)$. It is not difficult to see that the coefficient of the term \mathbf{x}^β in $\sum_{i=1}^s \text{Proj}(h_i, \sigma(T))^2$ equals that of the term \mathbf{x}^β in $\sum_{i=1}^s h_i^2$. Thus, \mathbf{x}^β does not appear in $p - \sum_{i=1}^s \text{Proj}(h_i, \sigma(T))^2$ since $p - \sum_{i=1}^s h_i^2 = 0$. \square

Theorem 3 Assume $p = \sum c_\alpha \mathbf{x}^\alpha$ is SOS, \mathbf{Q} satisfies $SOSS(p, \mathbf{Q})$ and $T \subseteq \mathcal{V}(p, \mathbf{Q})$. If $\psi_{\mathbf{Q}}(\alpha + \beta) \subseteq T$ for any $\alpha, \beta \in \sigma(T)$, then $p_1 = \sum_{\alpha \in \mathcal{S}(p), \psi_{\mathbf{Q}}(\alpha) \subseteq T} c_\alpha \mathbf{x}^\alpha$ is SOS.

Proof Suppose $p = \sum_{i=1}^s h_i^2$ and $p'_1 = p - p_1$. Set $h'_i = \text{Proj}(h_i, \sigma(T))$ and $h''_i = h_i - h'_i$, then $p = \sum_{i=1}^s (h'_i)^2 + 2 \sum_{i=1}^s h'_i h''_i + \sum_{i=1}^s (h''_i)^2$. By Lemma 5, $\beta \notin \mathcal{S}(p - \sum_{i=1}^s (h'_i)^2)$ for any $\beta \in \mathcal{S}(p_1)$, i.e., $\mathcal{S}(p_1) \cap \mathcal{S}(p - \sum_{i=1}^s (h'_i)^2) = \emptyset$. Since $\psi_{\mathbf{Q}}(\alpha + \beta) \subseteq T$ for any $\alpha, \beta \in \sigma(T)$, by the definition of $\sigma(T)$, $\psi_{\mathbf{Q}}(\beta) \subseteq T$ for any $\beta \in \mathcal{S}(\sum_{i=1}^s (h'_i)^2)$. Thus, $\mathcal{S}(p'_1) \cap \mathcal{S}(\sum_{i=1}^s (h'_i)^2) = \emptyset$. Summarizing the above, we have

1. $p_1 + p'_1 = \sum_{i=1}^s (h'_i)^2 + (p - \sum_{i=1}^s (h'_i)^2)$,
2. $\mathcal{S}(p_1) \cap \mathcal{S}(p - \sum_{i=1}^s (h'_i)^2) = \emptyset$, and
3. $\mathcal{S}(p'_1) \cap \mathcal{S}(\sum_{i=1}^s (h'_i)^2) = \emptyset$.

Therefore, $p_1 = \sum_{i=1}^s (h'_i)^2$. \square

Definition 8 (Split polynomial) Let \mathbf{Q} be a finite set satisfying $SOSS(p, \mathbf{Q})$ for a polynomial p . If there exist some pairwise disjoint nonempty subsets $T_i (i = 1, \dots, u)$ of $\mathcal{V}(p, \mathbf{Q})$ such that

1. $\psi_{\mathbf{Q}}(\alpha + \beta) \subseteq T_i$ for any $\alpha, \beta \in \sigma(T_i)$ (see Definition 7) for any $i = 1, \dots, u$, and
2. for any $\alpha \in \mathcal{S}(p)$, there exist exact one T_i such that $\psi_{\mathbf{Q}}(\alpha) \subseteq T_i$,

then p is said to be a *split polynomial* with respect to T_1, \dots, T_u .

Theorem 4 Suppose $p = \sum c_{\alpha} \mathbf{x}^{\alpha}$ is a split polynomial with respect to T_1, \dots, T_u , then p is SOS if and only if each $p_i = \sum_{\alpha \in \mathcal{S}(p), \psi_{\mathbf{Q}}(\alpha) \subseteq T_i} c_{\alpha} \mathbf{x}^{\alpha}$ is SOS for $i = 1, \dots, u$.

Proof Necessity is a direct corollary of Theorem 3. For sufficiency, note that the second condition of Definition 8 guarantees that $\mathcal{S}(p_i) \cap \mathcal{S}(p_j) = \emptyset$ for any $i \neq j$ and $p = \sum_{i=1}^u p_i$. \square

Now, we give the relation between convex cover polynomial and split polynomial, which indicates that split polynomial is a wider class of polynomials.

Theorem 5 If p is a convex cover polynomial, then p is a split polynomial. The converse is not true.

Proof If p is a convex cover polynomial then there exist pairwise disjoint faces $F_i (i = 1, \dots, u)$ of $\mathcal{N}(p)$ such that $\mathcal{S}(p) \subseteq \cup_{i=1}^u F_i$. Suppose $\text{conv}(\mathbf{Q} + \mathbf{Q}) = \mathcal{N}(p)$ and \mathbf{Q} satisfies $SOSS(p, \mathbf{Q})$. Let $T_i = \{\alpha \in \mathcal{V}(p, \mathbf{Q}) \mid 2\alpha \in F_i\}, i = 1, \dots, u$. We prove that p is a split polynomial with respect to T_1, \dots, T_u .

We claim that $\sigma(T_j) = \{\gamma \in \mathbf{Q} \mid 2\gamma \in F_j\}$ for $j = 1, \dots, u$. If there exist $\gamma_0 \in \sigma(T_j), 2\gamma_0 \notin F_j$, as F_j is a face of $\mathcal{N}(p)$, there exist a linear function π such that $\pi(2\gamma_0) > \pi(\alpha)$ for any $\alpha \in F_j$. By the Definition of $\psi_{\mathbf{Q}}(2\gamma_0)$, there exists $\beta_0 \in \psi_{\mathbf{Q}}(2\gamma_0) \subseteq T_j$ such that $\pi(2\beta_0) \geq \pi(2\gamma_0)$. This contradicts with $2\beta_0 \in F_j$. Thus, $\sigma(T_j) \subseteq \{\gamma \in \mathbf{Q} \mid 2\gamma \in F_j\}$.

We then prove that $\{\gamma \in \mathbf{Q} \mid 2\gamma \in F_j\} \subseteq \sigma(T_j)$. Assume that there exists $\gamma_0 \in \mathbf{Q}$ with $2\gamma_0 \in F_j$ such that $\gamma_0 \notin \sigma(T_j)$. Then there exists $\beta_0 \in \psi_{\mathbf{Q}}(2\gamma_0)$ such that $2\beta_0 \notin F_j$. Because F_j is a face of $\mathcal{N}(p)$, it is not difficult to see that if $\alpha_1 + \alpha_2 \in F_j$ where $\alpha_1, \alpha_2 \in \mathbf{Q}$, then $2\alpha_1 \in F_j, 2\alpha_2 \in F_j$. Therefore, $2\beta \in F_j$ for any $\beta \in \psi_{\mathbf{Q}}(2\gamma_0)$, which contradicts with $2\beta_0 \notin F_j$.

Now we have $\sigma(T_j) = \{\gamma \in \mathbf{Q} \mid 2\gamma \in F_j\}$. By Lemma 4, $\psi_{\mathbf{Q}}(\alpha + \beta) \subseteq T_j$ for any $\alpha, \beta \in \sigma(T_j)$. Since $\mathcal{S}(p) \subseteq \cup_{i=1}^u F_i$ and F_i are pairwise disjoint, there exists exact one T_i such that $\psi_{\mathbf{Q}}(\alpha) \subseteq T_i$ for any $\alpha \in \mathcal{S}(p)$. As a result, p is a split polynomial with respect to T_1, \dots, T_u .

Note that the Motzkin polynomial in Example 7 is a split polynomial but not a convex cover polynomial since $x_1^2 x_2^2$ lies in the interior of $\mathcal{N}(p)$ (see Figure 3). That means the converse is not true. \square

Remark 2 One may wonder under what condition a split polynomial is a convex cover polynomial. A reasonable conjecture may be as this:

Let \mathbf{Q} be a finite set satisfying $SOSS(p, \mathbf{Q})$ with $\text{conv}(\mathbf{Q} + \mathbf{Q}) = \mathcal{N}(p)$ for a polynomial p . If $\mathcal{V}(p, \mathbf{Q})$ contains only vertices of $\text{conv}(\mathbf{Q})$, then p is a split polynomial if and only if p is a convex polynomial.

Unfortunately, the conjecture is not true. For example, let

$$p = x_1^4 x_2^2 x_3^2 + x_1^2 x_2^4 x_3^2 - 2x_1^2 x_2^2 x_3^2 + x_3^2 + x_1^2 x_2^2 + x_1^2 x_2^2 x_3^4,$$

then

$$\begin{aligned}\mathbf{Q} &= \{(2, 1, 1), (1, 2, 1), (1, 1, 1), (0, 0, 1), (1, 1, 0), (1, 1, 2)\}, \\ \mathcal{V}(p, \mathbf{Q}) &= \{(2, 1, 1), (1, 2, 1), (0, 0, 1), (1, 1, 0), (1, 1, 2)\}.\end{aligned}$$

Obviously, $\mathcal{V}(p, \mathbf{Q})$ contains only vertices of $\text{conv}(\mathbf{Q})$. Set $T_1 = \{(2, 1, 1), (1, 2, 1), (0, 0, 1)\}$, $T_2 = \{(1, 1, 0), (1, 1, 2)\}$, then it is easy to check p is a split polynomial with respect to T_1, T_2 . But p is not a convex cover polynomial by Proposition 1 because $x_1^2 x_2^2 x_3^2$ lies in the interior of $\mathcal{N}(p)$.

The example indicates that the relation between split polynomial and convex cover polynomial may be complicated. We do not find a good sufficient condition for a split polynomial to be a convex cover polynomial.

5 Algorithm

Existing SDP based SOS solvers consists of the following two main steps: computing a set \mathbf{Q} which satisfies $\text{SOSS}(p, \mathbf{Q})$ for a given p ; solving the feasibility problem of (7) related to \mathbf{Q} by SDP solvers. In this section, we give a new algorithm (Algorithm 2) for SOS decomposition. The algorithm employs the following strategies. First, we give a different technique for computing an initial set \mathbf{Q} which satisfies $\text{SOSS}(p, \mathbf{Q})$ for a given p . Second, we check one necessary condition (Lemma 8) to refute quickly some non-SOS polynomials. Third, if the input polynomial is detected to be a split polynomial, we reduce the problem into several smaller sub-problems based on Theorem 4. This section is dedicated to describe the strategies in detail and the performance of the algorithm is reported in the next section.

We first describe the new technique for computing an initial set \mathbf{Q} . The following lemma is a direct corollary of the result in [28] (see also Eq. (4) in Section 2).

Lemma 6 *Suppose p is a polynomial and γ is a given vector. Let $c = \max_{\alpha \in \frac{1}{2}\mathbf{P}^e} \gamma^T \alpha$. For any \mathbf{Q} which satisfies $\text{SOSS}(p, \mathbf{Q})$, after deleting every β in \mathbf{Q} such that $\gamma^T \beta > c$, $\text{SOSS}(p, \mathbf{Q})$ still holds.*

By Lemma 6, it is easy to give a method for computing an initial set \mathbf{Q} which satisfies $\text{SOSS}(p, \mathbf{Q})$ for a given p . That is, first choose a coarse set \mathbf{Q} which satisfies $\text{SOSS}(p, \mathbf{Q})$, e.g., the set defined by Eq. (5); then prune the superfluous elements in \mathbf{Q} by choosing *randomly* γ . This is indeed a common method in existing work [21, 14, 32].

We employ a different strategy to construct an initial \mathbf{Q} satisfying $\text{SOSS}(p, \mathbf{Q})$. The procedure is as follows. For a given polynomial p , firstly, we compute the set $\frac{1}{2}\mathbf{P}^e$ (recall that $\mathbf{P}^e = \mathbf{P} \cap (2\mathbb{Z}_+^n)$ where \mathbf{P} is the support of p) and an over approximation set \mathbf{Q} of integer points in $\text{conv}(\frac{1}{2}\mathbf{P}^e)$. Secondly, let B be the matrix whose columns are all the vectors of $\frac{1}{2}\mathbf{P}^e$. We choose one by one the hyperplanes whose normal directions are the eigenvectors of BB^T to delete superfluous lattice points in \mathbf{Q} by Lemma 6.

Definition 9 We denote by $\text{PCAG}(p)$ the above procedure to compute an initial \mathbf{Q} satisfying $\text{SOSS}(p, \mathbf{Q})$ for a given polynomial p .

We cannot prove the above strategy is better in general than the random one. However, inspired by principal component analysis (PCA), we believe in many cases the shape of $\text{conv}(\frac{1}{2}\mathbf{P}^e)$ depends on eigenvectors of BB^T . On a group of randomly generated examples (see Example 4), we show that the size of \mathbf{Q} obtained by using random hyperplanes to delete superfluous lattice points are 10% greater than that of the output of our algorithm PCAG (see Figure 2).

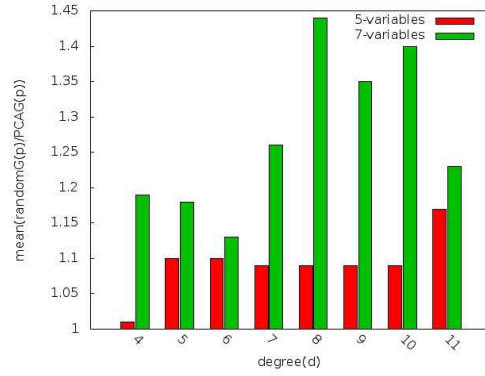


Fig. 2 Mean ratio of $\#(\mathbf{Q})$ between random algorithm and PCAG(p) on every random group $SQR(k, n, d, t)$. Red color corresponds to $k = 4, n = 5, t = 3$ and green color corresponds to $k = 5, n = 7, t = 4$. For any given (k, n, d, t) , we generate 10 polynomials randomly.

Example 4 $SQR(k, n, d, t) = g_1^2 + \dots + g_k^2$ where $\deg(g_i) = d, \#(\mathcal{S}(g_i)) = t, \#(\text{var}(g_i)) = n$.

Lemma 7 [11, 33] For a polynomial p and a set \mathbf{Q} which satisfy $SOSS(p, \mathbf{Q})$, after deleting every element α in \mathbf{Q} which satisfies that $2\alpha \notin \mathbf{P}^e$ and $\phi_{\mathbf{Q}}(2\alpha) = \{\alpha\}$, the relation $SOSS(p, \mathbf{Q})$ still holds.

Definition 10 We denote by $\text{EXACTG}(p)$ the procedure which deletes superfluous elements of the output of PCAG(p) based on Lemma 7.

The following lemma is a simple but very useful necessary condition which can detect non-SOS polynomials efficiently in many cases.

Lemma 8 Suppose \mathbf{Q} satisfies $SOSS(p, \mathbf{Q})$ for a polynomial p . If p is SOS, then $\alpha \in \mathbf{Q} + \mathbf{Q}$ for any $\alpha \in \mathcal{S}(p)$.

Proof If p is SOS, since p and \mathbf{Q} satisfy relation $SOSS(p, \mathbf{Q})$, there are q_1, \dots, q_s such that $p = \sum_{i=1}^s q_i^2$ and $\mathcal{S}(q_i) \subseteq \mathbf{Q}$. Hence, for every monomial \mathbf{x}^α of p there are $q_i, \mathbf{x}^\beta, \mathbf{x}^\gamma$ such that $\mathbf{x}^\beta, \mathbf{x}^\gamma$ are monomials of q_i and $\mathbf{x}^\alpha = \mathbf{x}^\beta \mathbf{x}^\gamma$. Therefore, $\alpha \in \mathbf{Q} + \mathbf{Q}$ for any $\alpha \in \mathcal{S}(p)$. \square

Example 5 [4] Let $q(x, y, z) = 1 + x^2y^2 + y^2z^2 + z^2x^2 - 4xyz$. It is easy to know that

$$\begin{aligned} \frac{1}{2}\mathbf{P}^e &= \{(0, 0, 0), (1, 1, 0), (1, 0, 1), (0, 1, 1)\}, \\ \mathbf{Q}^0 &= \{(0, 0, 0), (1, 0, 0), (0, 1, 0), (0, 0, 1), (1, 1, 0), (1, 0, 1), (0, 1, 1)\}, \end{aligned}$$

and $SOSS(q, \mathbf{Q}^0)$ holds. By Lemma 7, after deleting $(1, 0, 0), (0, 1, 0), (0, 0, 1)$ from \mathbf{Q}^0 , we have $\mathbf{Q} = \text{EXACTG}(q) = \{(0, 0, 0), (1, 1, 0), (1, 0, 1), (0, 1, 1)\}$ and $SOSS(q, \mathbf{Q})$ holds. Since $(1, 1, 1) \notin \mathbf{Q} + \mathbf{Q}$, by Lemma 8, p is not SOS.

For an input polynomial p , by setting $\mathbf{Q} = \text{EXACTG}(p)$, we obtain a set \mathbf{Q} satisfying $SOSS(p, \mathbf{Q})$. Now, we check whether or not p is a split polynomial related to this \mathbf{Q} . And if it is, the original problem can be reduced to several smaller sub-problems. The details are described formally as Algorithm 1 and Algorithm 2.

Algorithm 1. MonomialRelation

Input: $p \in \mathbb{Q}[\mathbf{x}]$
Output: The map $\psi_{\mathbf{Q}}$ defined by Definition 6

```

1  $\mathbf{Q} = \text{EXACTG}(p)$ ;
2 Let  $C$  be a map from  $\mathbf{Q}$  to  $\{\text{true}, \text{false}\}$ ;
3 for  $\alpha \in \mathbf{Q}$  do  $C(\alpha) = \text{false}$ ;
4 Let  $\mathcal{V}(p, \mathbf{Q})$  be the set defined by Definition 5;
5 Initialize  $\psi_{\mathbf{Q}}(\alpha) = \emptyset$  for any  $\alpha \in \mathbb{Z}_+^n$ ;
6 for  $\alpha \in \mathcal{V}(p, \mathbf{Q})$  do
7    $\psi_{\mathbf{Q}}(2\alpha) = \{\alpha\}$ ;
8    $C(\alpha) = \text{true}$ ;
9 Let  $\text{run} = \text{true}$ ;
10 while  $\text{run}$  do
11    $\text{run} = \text{false}$ ;
12   for  $\alpha \in \mathbf{Q}$  do
13     if  $C(\alpha)$  then
14        $C(\alpha) = \text{false}$ ;
15       for  $\beta \in \mathbf{Q}$  do
16         if  $\psi_{\mathbf{Q}}(2\alpha) \not\subseteq \psi_{\mathbf{Q}}(\alpha + \beta)$  then
17            $\psi_{\mathbf{Q}}(\alpha + \beta) = \psi_{\mathbf{Q}}(\alpha + \beta) \cup \psi_{\mathbf{Q}}(2\alpha)$ ;
18           if  $\alpha + \beta \in 2\mathbb{Z}_+^n$  then
19              $\text{run} = \text{true}; C((\alpha + \beta)/2) = \text{true}$ ;
20 return  $\psi_{\mathbf{Q}}$ ;

```

Algorithm 2. QuickSOS

Input: $p \in \mathbb{Q}[\mathbf{x}]$
Output: **false** that means p is not SOS; or $\{q_1, \dots, q_s\}$ where p, q_i satisfy Eq. (2) numerically

```

1 Let  $\psi_{\mathbf{Q}}$  be the output of MonomialRelation( $p$ );
2 for  $\alpha \in S(p)$  do if  $\alpha \notin \mathbf{Q} + \mathbf{Q}$  then return false; // Lemma 8
3 for  $\alpha \in S(p)$  do
4   if  $p$  is a split polynomial with respect to  $\psi_{\mathbf{Q}}(\alpha)$  then
5     Let  $p_1, p_2$  be as in Theorem 4;
6     Let  $R_1$  be the output of QuickSOS( $p_1$ );
7     Let  $R_2$  be the output of QuickSOS( $p_2$ );
8     if  $R_1$  or  $R_2$  is false then return false;
9     return  $R_1 \cup R_2$ ;
10 return SOS( $p, \mathbf{Q}$ ); // Notation 1

```

Example 6 We illustrate QuickSOS on the polynomial p in Example 3. First,

$$\begin{aligned}
S(p) &= \{(0, 6), (6, 0), (0, 4), (4, 0), (2, 2)\}, \\
\mathbf{Q} &= \{(0, 2), (0, 3), (1, 1), (1, 2), (2, 0), (2, 1), (3, 0)\}, \\
\mathcal{V}(p, \mathbf{Q}) &= \{(0, 2), (0, 3), (2, 0), (3, 0)\}.
\end{aligned}$$

Second, $\psi_{\mathbf{Q}}((0, 4)) = \{(0, 2)\}$, $\psi_{\mathbf{Q}}((0, 6)) = \{(0, 3)\}$, $\psi_{\mathbf{Q}}((4, 0)) = \{(2, 0)\}$, $\psi_{\mathbf{Q}}((6, 0)) = \{(3, 0)\}$, $\psi_{\mathbf{Q}}((2, 2)) = \{(0, 2), (2, 0)\}$. Set $T = \psi_{\mathbf{Q}}((2, 2)) = \{(2, 0), (0, 2)\}$, it is easy to see that p is a split polynomial with respect to T and $p_1 = x_1^4 - 2x_1^2x_2^2 + x_2^4$, $p_2 = x_1^6 + x_2^6$.

Third, similarly, $\text{QuickSOS}(p_2)$ divides p_2 into $p_{21} = x_1^6, p_{22} = x_2^6$. Finally, $\text{QuickSOS}(p)$ outputs “ $1.0000 * x_2^4 + 1.0000 * x_2^6 - 2.0000 * x_1^2 * x_2^2 + 1.0000 * x_1^4 + 1.0000 * x_1^6 = (-1.00 * x_2^2 + 1.00 * x_1^2)^2 + (1.00 * x_2^3)^2 + (1.00 * x_1^3)^2$ ”.

Example 7 (Motzkin polynomial) $f = x_1^4 x_2^2 + x_1^2 x_2^4 - 3x_1^2 x_2^2 + 1$.

Because $\mathcal{S}(f) = \{(4, 2), (2, 2), (2, 4), (0, 0)\}$ and $\mathbf{Q} = \{(0, 0), (1, 1), (2, 1), (1, 2)\}$, $\text{MonomialRelation}(f)$ returns $\psi_{\mathbf{Q}}((4, 2)) = \{(2, 1)\}$, $\psi_{\mathbf{Q}}((2, 4)) = \{(2, 4)\}$, $\psi_{\mathbf{Q}}((2, 2)) = \{(1, 1)\}$, $\psi_{\mathbf{Q}}((0, 0)) = \{(0, 0)\}$. Then $\text{QuickSOS}(f)$ will return false when it reaches line 8 for $\alpha = (2, 2)$.

Remark 3 Let $\mathbf{Q} = \text{EXACTG}(p)$. By Definition 8, to determine whether p is a split polynomial, one should check all the non-empty subsets of $\mathcal{V}(p, \mathbf{Q})$. However, this approach is obviously inefficient. Therefore, in Algorithm 2 we only check whether p is a split polynomial with respect to $\psi_{\mathbf{Q}}(\alpha)$ for some $\alpha \in \mathcal{S}(p)$. Although this incomplete check may miss some split polynomials, as is shown in the next section, it is effective in many cases.

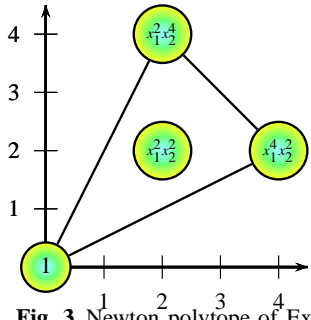


Fig. 3 Newton polytope of Example 7.

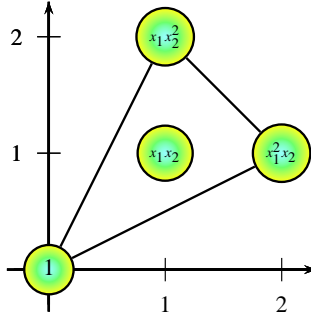


Fig. 4 \mathbf{Q}^0 of Example 7.

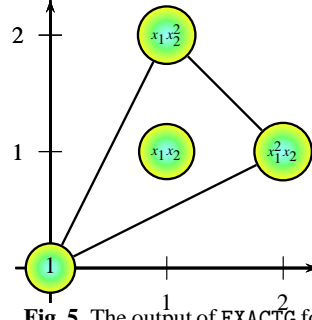


Fig. 5 The output of EXACTG for Example 7.

6 Experiments

The above algorithms have been implemented as a C++ program, QuickSOS . Compilation has been done using g++ version 4.6.3 with optimization flags -O2. We use Singular [7] to read polynomials from files or standard input and use Csdp [3] as SDP solver. The program has been tested on many benchmarks in the literature and on lots of examples generated randomly.

We report in this section corresponding experimental data of our program and some well-known SOS solvers, such as YALMIP, SOSTOOLS, SOSOPT. The matlab version is R2011b and SOSTOOLS's version is 3.00. Both YALMIP and SOSOPT are the latest release. The SDP solver of YALMIP, SOSTOOLS and SOSOPT is SeDuMi 1.3.

All the numerical examples listed were computed on a 64-bit Intel(R) Core(TM) i5 CPU 650 @ 3.20GHz with 4GB RAM memory and Ubuntu 12.04 GNU/Linux.

6.1 Examples

In this subsection, we define four classes of examples. The first class of examples are modified from [22], which are positive but not necessarily SOS. The second one is from [8, 10].

The other two classes are sparse polynomials randomly generated by Maple's command `randpoly` where the third class of polynomials are constructed in the form of SOS.

The number of elements in a set \mathcal{Q} is denoted by $\#(\mathcal{Q})$, $\deg(p)$ denotes the total degree of a polynomial p , $\text{var}(p)$ denotes the set of variables occurring in a polynomial p .

6.1.1 B_m

$B_m = (\sum_{i=1}^{3m+2} x_i^2)^2 - 2 \sum_{i=1}^{3m+2} x_i^2 \sum_{j=1}^m x_{i+3j+1}^2$, where $x_{3m+2+r} = x_r$. B_m is modified from [22]. For any $m \in \mathbb{Z}_+$, B_m is homogeneous and is a positive polynomial.

6.1.2 $p_{i,j}$

Monotone Column Permanent (MCP) Conjecture was given in [8]. When $n = 4$, this Conjecture is equivalent to decide whether $p_{12}, p_{13}, p_{22}, p_{23}$ are positive polynomials and this case has been studied in [10].¹

6.1.3 $SQR(k, n, d, t)$ (see Example 4)

$SQR(k, n, d, t) = g_1^2 + \dots + g_k^2$ where $\deg(g_i) = d$, $\#(S(g_i)) = t$, $\#(\text{var}(g_i)) = n$.

6.1.4 $RN(n, d)$

$RN(n, d) = g_1^2 + g_2 \sum_{i=1}^n x_i + 100g_3^2 + 100$, where $\deg(g_1) = d$, $\deg(g_2) = d - 3$, $\deg(g_3) = d - 2$, $\text{var}(g_i) = \{x_1, \dots, x_n\}$. For any given (n, d) where $n \in \{5, 10\}$ and $4 \leq d \leq 12$, we generate 10 corresponding polynomials.

6.2 Results

If we only compare the timings of different tools, the comparison is somehow unfair since the implementation languages are different. Since the main idea of this paper is to compute smaller set \mathbf{Q} for given polynomial p which make relation $\text{SOSS}(p, \mathbf{Q})$ hold, we also report the comparison of the size of \mathbf{Q} computed by different tools. It is reasonable to believe that the total time of computing SOS decomposition becomes shorter as the size of \mathbf{Q} getting smaller if we use the same SDP solver and the cost of computing smaller \mathbf{Q} is not expensive. In fact, for all the following examples except B_m , the time taken in computing \mathbf{Q} by QuickSOS is less than 0.1 seconds.

We explain the notations in the following tables. Each (b, s) for QuickSOS's $\#(\mathbf{Q})$ means QuickSOS divides the polynomial into b polynomials p_1, \dots, p_b and s is the largest number of $\#(\mathbf{Q}_i)$ corresponding to p_i . The “—” denotes that there is no corresponding output.

The results on B_m by these tools are listed in Table 1. The polynomials B_1 and B_2 are SOS, the others are not. All the above tools except SOSOPT give correct² outputs on B_m . Although B_i is not a sparse polynomial, our algorithm can also reduce $\#(\mathbf{Q})$. When the size of polynomial is large, SOSOPT takes so much time to solve it. This phenomenon also occurs in the following examples. For convenience, we do not list the results of SOSOPT in the following.

Table 1 The results on B_m .

Tools	#(\mathbf{Q})					time(s)				
	B_1	B_2	B_3	B_4	B_5	B_1	B_2	B_3	B_4	B_5
QuickSOS	(5,1)	(1,33)	(1,55)	(1,94)	(1,150)	0.00	0.15	0.62	6.79	73.27
YALMIP	15	36	66	105	153	0.24	0.53	0.72	1.26	167.21
SOSTOOLS	15	36	66	105	153	0.30	0.42	2.09	21.32	163.05
SOSOPT	15	36	—	—	—	0.25	3.01	error	error	error

Table 2 The results on $p_{i,j}$.

Tools	#(\mathbf{Q})				time(s)			
	$p_{1,2}$	$p_{1,3}$	$p_{2,2}$	$p_{2,3}$	$p_{1,2}$	$p_{1,3}$	$p_{2,2}$	$p_{2,3}$
QuickSOS	(1,77)	(5,15)	(1,62)	(6,39)	1.98	0.01	1.25	0.19
YALMIP	77	29	62	53	4.93	1.81	4.97	4.10
SOSTOOLS	wrong	wrong	62	wrong	wrong	wrong	3.77	wrong

The results on $p_{i,j}$ by those tools are listed in Table 2.

Table 3 lists the results on examples SQR (see Example 4). We randomly generate 10 polynomials for every (k, n, d, t) . All the outputs of QuickSOS and YALMIP are correct. Some data corresponding to SOSTOOLS are “wrong”, which means that SOSTOOLS’s output is wrong or there is an error occurred during its execution. For many examples of SQR , QuickSOS can divide the original polynomial into some simpler polynomials. By the complexity analysis in Section 2, this division can greatly improve efficiency.

We demonstrate this fact by one polynomial of group $SQR(4, 5, 10, 3)$.

Example 8 $p = (-91w^4x^2yz^3 - 41k^4xy^2z^2 - 14kw^3y^2z)^2 + (-40kx^7yz + 16w^4xy + 65w^2y^4)^2 + (11kx^2y^6z - 34k^5x^3z - 18kyz^5)^2 + (-26k^4w^3xyz - 35xy^6z^3 - 57kw^2x^2z^3)^2$.

Remark 4 As explained before, SQR is constructed in the form of SOS. But the polynomial is expanded before input to the tools.

In Example 8, QuickSOS divides p into four simpler polynomials p_1, p_2, p_3, p_4 . For each simpler polynomial p_i , QuickSOS constructs a set \mathbf{Q}_i whose size is 3 and $SOSS(p_i, \mathbf{Q}_i)$ holds. YALMIP constructs one \mathbf{Q} for p whose size is 97 and SOSTOOLS also constructs one \mathbf{Q} for p whose size is 104. If the time consumed by constructing \mathbf{Q} is short compared with total time and assume these three tools use the same SDP solver, the ratio of total time of three tools is $4(3^{2+2c}) : 97^{2+2c} : 104^{2+2c}$ where $1 \leq c \leq 2$. In fact, in our experiments, the total time of these three tools on this example is 0.02 seconds, 23.91 seconds and 48.47 seconds, respectively.

In addition to efficiency, correction is also important. Figure 6 shows the number of “wrong” of SOSTOOLS on every group of random polynomials SQR . As explained above, “wrong” means that SOSTOOLS’s output is wrong or there is an error occurred during its execution. Those “wrong”s are caused by numerical instability. Therefore, the number of “wrong” increases with the increase of the problem’s size.

The above experiments are all about polynomials which are SOS. Figure 7 is about timings for refuting non-SOS polynomials. For all 180 RN polynomials, QuickSOS takes 1.07 seconds to refute all of them. And there are polynomials in these 180 polynomials on

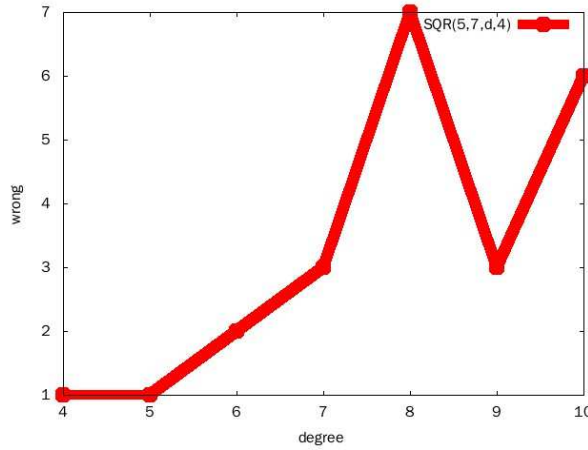
¹ The polynomials can be found at http://www4.ncsu.edu/~kaltofen/software/mcp_conj_4/.

² The meaning of correction is that the output is right with respect to a certain numerical error.

Table 3 $\#(\mathbf{Q})$ of random polynomials $SQR(k, n, d, t)$

Tools	1	2	3	4	5	6	7	8	9	10
$\#(\mathbf{Q})$										
$k = 4, n = 5, d = 5, t = 3$										
QuickSOS	(2,15)	(1,44)	(2,11)	(4,4)	(4,4)	(1,25)	(4,5)	(3,9)	(2,8)	(2,20)
YALMIP	24	45	33	18	23	36	22	20	15	25
SOSTOOLS	24	45	33	18	23	36	22	20	15	25
$k = 4, n = 5, d = 10, t = 3$										
QuickSOS	(4,3)	(4,3)	(4,10)	(3,6)	(2,7)	(4,4)	(4,3)	(4,3)	(4,3)	(2,26)
YALMIP	97	91	42	23	45	40	101	62	95	52
SOSTOOLS	104	94	36	23	48	41	109	70	104	52
$k = 5, n = 7, d = 5, t = 4$										
QuickSOS	(4,7)	(5,5)	(2,13)	(5,4)	(4,11)	(5,4)	(4,7)	(3,12)	(5,5)	(4,10)
YALMIP	21	33	24	24	28	24	21	28	42	33
SOSTOOLS	wrong	33	24	24	28	24	21	28	42	33
$k = 5, n = 7, d = 10, t = 4$										
QuickSOS	(5,4)	(5,4)	(5,5)	(5,5)	(5,4)	(5,4)	(5,4)	(5,4)	(3,11)	(5,4)
YALMIP	45	82	74	59	48	70	79	63	41	57
SOSTOOLS	wrong	wrong	wrong	63	57	76	wrong	67	wrong	wrong
$k = 5, n = 7, d = 5, t = 6$										
QuickSOS	(1,26)	(1,29)	(1,28)	(1,72)	(1,37)	(1,30)	(1,27)	(4,7)	(2,14)	(1,61)
YALMIP	28	38	28	82	48	31	33	34	34	69
SOSTOOLS	wrong	38	28	82	wrong	31	33	wrong	34	wrong
$k = 5, n = 7, d = 8, t = 6$										
QuickSOS	(4,7)	(4,6)	(4,7)	(4,7)	(4,6)	(4,6)	(4,6)	(2,24)	(4,6)	(4,6)
YALMIP	38	34	71	121	51	57	75	100	47	29
SOSTOOLS	39	wrong	wrong	128	67	67	78	111	52	31

which SOSTOOLS cannot finish execution within 10000 seconds. So we do not list its output. Figure 7 is the mean time of YALMIP for every group of polynomials.

**Fig. 6** The number of “wrong” of SOSTOOLS on every group of random polynomials.

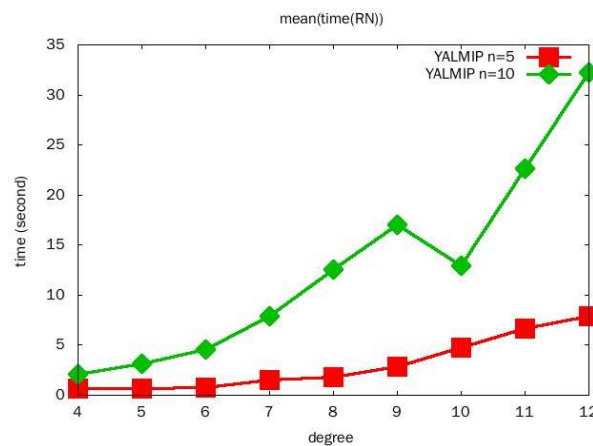


Fig. 7 Mean running time of YALMIP on every group of RN polynomials.

Acknowledgements

The authors would like to thank Prof. Hoon Hong for his suggestion and discussion on an early version of this paper and thank Xiaojuan Wu for helping improve the presentation of the paper.

The authors are very much grateful to the reviewer whose constructive suggestions and insightful comments not only help improve greatly the presentation of the paper but also lead to some new results.

References

1. Artin, E.: *er die zerlegung definiter funktionen in quadrate*. Abhandlungen aus dem Mathematischen Seminar der Universitt Hamburg **5**(1), 100–115 (1927). DOI 10.1007/BF02952513. URL <http://dx.doi.org/10.1007/BF02952513>
2. Blekherman, G.: There are significantly more nonnegative polynomials than sums of squares. Israel Journal of Mathematics **153**(1), 355–380 (2006)
3. Borchers, B.: Csdp, a c library for semidefinite programming. Optimization Methods and Software **11**(1–4), 613–623 (1999)
4. Choi, M.D., Lam, T.Y.: Extremal positive semidefinite forms. Mathematische Annalen **231**(1), 1–18 (1977)
5. Choi, M.D., Lam, T.Y., Reznick, B.: Sums of squares of real polynomials. In: Proceedings of symposia in pure mathematics, vol. 58, pp. 103–126. American Mathematical Society (1995)
6. Dai, L., Gan, T., Xia, B., Zhan, N.: Barrier certificates revisited. arXiv preprint arXiv:1310.6481 (2013)
7. Greuel, G.M., Pfister, G.: Singular and applications. Jahresbericht der **1505**(108), 4 (2006)
8. Haglund, J., Ono, K., Wagner, D.G.: Theorems and conjectures involving rook polynomials with real roots. In: in Proc. Topics in Number Theory and Combinatorics, State, pp. 207–221 (1997)
9. Kaltofen, E., Li, B., Yang, Z., Zhi, L.: Exact certification of global optimality of approximate factorizations via rationalizing sums-of-squares with floating point scalars. In: Proceedings of the twenty-first international symposium on Symbolic and algebraic computation, pp. 155–164. ACM (2008)
10. Kaltofen, E., Yang, Z., Zhi, L.: A proof of the monotone column permanent (mcp) conjecture for dimension 4 via sums-of-squares of rational functions. In: Proceedings of the 2009 Conference on Symbolic Numeric Computation, SNC '09, pp. 65–70. ACM, New York, NY, USA (2009)
11. Kim, S., Kojima, M., Waki, H.: Generalized lagrangian duals and sums of squares relaxations of sparse polynomial optimization problems. SIAM Journal on Optimization **15**(3), 697–719 (2005)
12. Kojima, M., Kim, S., Waki, H.: Sparsity in sums of squares of polynomials. Mathematical Programming **103**(1), 45–62 (2005)

13. Lasserre, J.B.: Global optimization with polynomials and the problem of moments. *SIAM Journal on Optimization* **11**(3), 796–817 (2001)
14. Lofberg, J.: Yalmip: A toolbox for modeling and optimization in matlab. In: *Computer Aided Control Systems Design, 2004 IEEE International Symposium on*, pp. 284–289. IEEE (2004)
15. Lofberg, J.: Pre-and post-processing sum-of-squares programs in practice. *Automatic Control, IEEE Transactions on* **54**(5), 1007–1011 (2009)
16. Matsukawa, Y., Yoshise, A.: A primal barrier function phase i algorithm for nonsymmetric conic optimization problems. *Japan Journal of Industrial and Applied Mathematics* **29**(3), 499–517 (2012)
17. Monteiro, R.: Primal–dual path-following algorithms for semidefinite programming. *SIAM Journal on Optimization* **7**(3), 663–678 (1997)
18. Motzkin, T.S.: The arithmetic-geometric inequality. *Inequalities (Proc. Sympos. Wright-Patterson Air Force Base, Ohio, 1965)* pp. 205–224 (1967)
19. Nesterov, Y.: Squared functional systems and optimization problems. In: *High performance optimization*, pp. 405–440. Springer (2000)
20. Nesterov, Y., Nemirovskii, A., Ye, Y.: Interior-point polynomial algorithms in convex programming, vol. 13. SIAM (1994)
21. Papachristodoulou, A., Anderson, J., Valmorbida, G., Prajna, S., Seiler, P., Parrilo, P.A.: SOS-TOOLS: Sum of squares optimization toolbox for MATLAB. <http://arxiv.org/abs/1310.4716> (2013). Available from <http://www.eng.ox.ac.uk/control/sostools>, <http://www.cds.caltech.edu/sostools> and <http://www.mit.edu/~parrilo/sostools>
22. Parrilo, P.A.: Structured semidefinite programs and semialgebraic geometry methods in robustness and optimization. Ph.D. thesis, California Inst. of Tech. (2000)
23. Parrilo, P.A.: Semidefinite programming relaxations for semialgebraic problems. *Mathematical Programming* **96**, 293–320 (2003)
24. Parrilo, P.A., Sturmfels, B.: Minimizing polynomial functions. *Algorithmic and quantitative real algebraic geometry, DIMACS Series in Discrete Mathematics and Theoretical Computer Science* **60**, 83–99 (2003)
25. Permenter, F., Parrilo, P.A.: Selecting a monomial basis for sums of squares programming over a quotient ring. In: *CDC*, pp. 1871–1876 (2012)
26. Powers, V., Wörmann, T.: An algorithm for sums of squares of real polynomials. *Journal of pure and applied algebra* **127**(1), 99–104 (1998)
27. Renegar, J.: A mathematical view of interior-point methods in convex optimization, vol. 3. Siam (2001)
28. Reznick, B.: Extremal PSD forms with few terms. *Duke Mathematical Journal* **45**, 363–374 (1978)
29. Reznick, B.: Forms derived from the arithmetic-geometric inequality. *Mathematische Annalen* **283**(3), 431–464 (1989)
30. Reznick, B.: Uniform denominators in hilbert’s seventeenth problem. *Mathematische Zeitschrift* **220**(1), 75–97 (1995)
31. Schweighofer, M.: Optimization of polynomials on compact semialgebraic sets. *SIAM Journal on Optimization* **15**(3), 805–825 (2005)
32. Seiler, P.: Sosopt: A toolbox for polynomial optimization. arXiv preprint arXiv:1308.1889 (2013)
33. Seiler, P., Zheng, Q., Balas, G.: Simplification methods for sum-of-squares programs. arXiv preprint arXiv:1303.0714 (2013)
34. Shor, N.Z.: Nondifferentiable optimization and polynomial problems, vol. 24. Springer (1998)
35. Sturm, J.F., Zhang, S.: On the long-step path-following method for semidefinite programming. *Operations Research Letters* **22**(4?), 145 – 150 (1998)
36. Sturmfels, B.: Polynomial equations and convex polytopes. *The American Mathematical Monthly* **105**(10), pp. 907–922 (1998)
37. Todd, M.J.: Semidefinite optimization. *Acta Numerica 2001* **10**, 515–560 (2001)
38. Vandenberghe, L., Boyd, S.: Semidefinite programming. *SIAM Review* **38**(1), 49–95 (1996)
39. Waki, H., Kim, S., Kojima, M., Muramatsu, M.: Sums of squares and semidefinite program relaxations for polynomial optimization problems with structured sparsity. *SIAM Journal on Optimization* **17**(1), 218–242 (2006)
40. Waki, H., Muramatsu, M.: A facial reduction algorithm for finding sparse sos representations. *Operations Research Letters* **38**(5), 361–365 (2010)
41. Wolkowicz, H., Saigal, R., Vandenberghe, L.: Handbook of semidefinite programming: theory, algorithms, and applications, vol. 27. Springer (2000)